RELEASE NOTES

GNU Toolchain for Atmel ARM Embedded Processors

Introduction

The Atmel ARM GNU Toolchain (5.3.1.487) supports Atmel ARM® devices. The ARM toolchain is based on the free and open-source GCC. This toolchain is built from sources published by ARM's "GNU Tools for ARM Embedded Processors" project at launchpad.net (https://launchpad.net/gcc-arm-embedded). The toolchain includes compiler, assembler, linker, binutils (GCC and binutils), GNU Debugger (GDB with builtin simulator) and Standard C library (newlib, newlib nano).

Table of Contents

1. Supported Configuration

1.1 Supported Hosts

This release includes the following:

- Bare metal EABI pre-built binaries for running on a Windows host
- Bare metal EABI pre-built binaries for running on a Linux host
- Bare metal EABI pre-built binaries for running on a Mac OS X host

1.2 Supported Targets

- Bare metal ARM EABI only (Use rdimon specs for semi-hosting enviroment)

2. Downloading, Installing, and Upgrading

The ARM GNU toolchain provided by Atmel is available for download. Use one of the following ways for installation.

2.1 Downloading/Installing on Windows

- to try the Atmel ARM GNU toolchain alone, you can download it from here[1]
- If you want to try the Atmel ARM GNU Toolchain along with Atmel studio, you can download and install Atmel Studio version 6.0 or later which will also install the Atmel ARM GNU toolchain. See Atmel studio release notes for more details.

2.2 Downloading/Installing on Linux

For Linux, the Atmel ARM GNU Toolchain is available as tar.gz archive which can be extracted using the tar utility. In order to install, simply extract to the location, from where you want to execute. The Linux builds are available here[2].

Note 64-bit version of libncurses and libc are required to run the tools.

2.3 Downloading/Installing on Mac OS

For Mac, the Atmel ARM GNU Toolchain is available as tar.gz archive which can be extracted using the tar utility. To install, extract to the location, from where you want to execute. MAC builds are available here[3]

2.4 Upgrading

If the Atmel ARM GNU Toolchain is installed by Atmel Studio installation, refer Atmel Studio documentation for more details.

If the toolchain is installed separately using one of the (Windows, Linux, Mac) installers, upgrading is not supported. You can install the new package side-by-side of the old package and use it.

[1] http://www.atmel.com/tools/atmel-arm-toolchain.aspx
[2] http://www.atmel.com/tools/atmel-arm-toolchain.aspx
[3] http://www.atmel.com/tools/atmel-arm-toolchain.aspx

3. Layout and Components

Listed below are some of the directories that you might want to look, to have a high level understanding of what is packaged inside the Atmel ARM GNU Toolchain. The layout is identical in Windows, Linux and Mac OS X.

3.1 Layout

The layout of the installation is as follows.

- **INSTALLDIR**

  The directory where the ARM GNU Toolchain is installed in the target machine.

- **INSTALLDIR\bin**

  The ARM software development programs. This directory should be in your PATH environemnt variable. (Note : If you are using this toolchain from within Atmel Studio, please configure Atmel studio appropriately). This includes

  - GNU Binutils
  - GCC
  - GDB

- **INSTALLDIR\arm-none-eabi\lib**

  The directory which have the ARM newlib libraries, startup files and linker scripts.

- **INSTALLDIR\arm-none-eabi\include**

  ARM-newlib header files. This is where the system include files will be searched for by the toolchain.

- **INSTALLDIR\lib**

  GCC libraries, other libraries and headers.

- **INSTALLDIR\libexec**

  GCC program components.

3.2 Components

The components used to build this toolchain along with their version number can be found here[1].

[1] http://distribute.atmel.no/tools/opensource/Atmel-ARM-GNU-Toolchain/5.3.1

4. Toolset Background

ARM GNU toolchain is a collection of executable software development tools for the Atmel ARM processors. These software development tools include:

1. Compiler
2. Assembler
3. Linker
4. Archiver
5. File converter
6. Other file utilities
7. C Library
8. Debugger

4.1 Compiler

The compiler is the GNU compiler collection, or GCC. This compiler is incredibly flexible and can be hosted on many platforms, it can target many different processors/operating systems(backends), and can be configured for multiple different languages (frontends).

The GCC included is targeted for the ARM processor, and is configured to compile C, and C++.

Because this GCC is targeted for the ARM, the main executable that is created is prefixed with the target name: `arm-none-eabi-gcc`. It is also referred to as ARM GCC.

`arm-none-eabi-gcc` is just a driver program. The compiler itself is called cc1.exe for C, or cc1plus.exe for C ++. Also the preprocessor cpp.exe will usually automatically be prefixed with the target name arm-none-eabi-cpp.exe. The actual set of component programs called is usually derived from the suffix of each soruce code file being processed.

GCC compiles a high-level computer lanugage into assembly, and that is all. It cannot work alone. GCC is coupled with another project, GNU Binutils, which provides the assembler, linker, librarian and more. Since GCC is just a driver program, it can automatically call the assembler and linker directly to build the final program.

4.2 Assembler, Linker, Librarian

GNU Binutils is a collection of binary utilities. This also includes the assembler,**as**. Sometimes you will see it referenced as GNU as or **gas**. Binutils includes the linker, **ld**; the librarian or archiver, **ar**. There are many other programs included that provide various functionality.

Binutils is configured for the ARM target and each of the programs is prefixed with the target name. So you have programs such as:

- **arm-none-eabi-as**: The GNU Assembler
- **arm-none-eabi-ld**: The GNU Linker
- **arm-none-eabi-ar**: The GNU Archiver, Create, modify, and extract from archives (libraries)
- **arm-none-eabi-ranlib**:Generate index of archive (library) contents
- **arm-none-eabi-objcopy**:Copy and translate object files
- **arm-none-eabi-objdump**:Display information from object files including disassembly
- **arm-none-eabi-size**:List section size, total size
- **arm-none-eabi-nm**:List symbol from object files.
- **arm-none-eabi-strings**:List printable strings from files
- **arm-none-eabi-strip**:Discard symbols

- **arm-none-eabi-readelf**:Display the contents of ELF file formats
- **arm-none-eabi-addr2line**:Convert addresses to file and line
- **arm-none-eabi-c++filt**:Filter to demangle encoded C++ symbols
- **arm-none-eabi-gdb**:Debugger to debug the target

See the binutils user manual for more information on what each program can do.

4.3 C Library

Newlib is the Standard C Library for ARM GCC. Newlib is the C library intended for use on embedded systems. It is a conglomeration of sevaral library parts. The library is ported to support ARM processor.

In addition to standard C library, newlib-nano also added to the toolchain package. Newlib-nano is newlib branch optimized for code size by ARM (https://launchpad.net/gcc-arm-embedded). To use newlib-nano, users should provide additional gcc link option "--specs=nano.specs". For more details, refer to the readme from here[1].

4.4 Debugging

- The toolchain distribution ships the `arm-none-eabi-gdb` which can be used for debugging purposes.
- Atmel Studio provides faclities to debug the executable produced by this toolchain. Note that `Atmel Studio` is currently free to the public, but it is not Open Source.

4.5 Source Code

This toolchain is built using the source from ARM's gcc-arm-embedded project 5-2016-q1-update[2] release. For Atmel's modification on source and build scripts, refer SOURCES.README from here[3].

[1] http://distribute.atmel.no/tools/opensource/Atmel-ARM-GNU-Toolchain/5.3.1
[2] https://launchpad.net/gcc-arm-embedded/5.0/5-2016-q1-update
[3] http://distribute.atmel.no/tools/opensource/Atmel-ARM-GNU-Toolchain/5.3.1

5. New and Noteworthy

Read ARM's gcc-arm-embedded project 5-2016-q1-update[1] release for updates and fixes. This section lists Atmel's modifications to that release.

- Default debug information is set to DWARF-2, which is supported by Atmel software debugger tools.
- Added object file wise memory usage details to map file. This shall be enabled using '--detailed-mem-usage' linker option.
- Multilib for armv7-a architecture with float variants Neon-vfpv4 and vfpv4-d16 FPUs.

Please read section "Architecture options usage" of gcc-arm-embedded project's readme (also available here)[2] for more information about multilib selections. Please refer below table for Atmel's modification to armv7-a multilibs.

Table 5-1.

ARM Core	Command Line Options	multilib
armv7-a thumb mode and Soft/ Softfp float-abi	-march=armv7-a -mthumb -mfloat-abi=soft -mfpu=vfpv4-d16	armv7-a/thumb-softfp-vfpv4-d16
armv7-a arm mode and soft/softfp float-abi	-march=armv7-a -mfloat-abi=soft -mfpu=vfpv4-d16	armv7-a/arm-softfp-vfpv4-d16
armv7-a thumb mode, hard float-abi, neon-vfpv4	-march=armv7-a -mthumb -mfloat-abi=hard -mfpu=neon-vfpv4	armv7-a/thumb-neon-vfpv4
armv7-a arm mode, hard float-abi, neon-vfpv4	-march=armv7-a -mfloat-abi=hard -mfpu=neon-vfpv4	armv7-a/arm-neon-vfpv4
armv7-a thumb mode, hard float-abi, vfpv4-d16	-march=armv7-a -mthumb -mfloat-abi=hard -mfpu=vfpv4-d16	armv7-a/thumb-vfpv4-d16
armv7-a arm mode, hard float-abi, vfpv4-d16	-march=armv7-a -mfloat-abi=hard -mfpu=vfpv4-d16	armv7-a/arm-vfpv4-d16

5.1 Supported Architectures

armv2	armv5e	armv6z	armv8-a+crc
armv2a	armv5t	armv6zk	armv8-m.base
armv2	armv5te	armv7	armv8-m.main
armv2a	armv6	armv7-a	armv8-m.main+dsp
armv3	armv6-m	armv7-m	iwmmxt
armv3m	armv6j	armv7-r	iwmmxt2
armv4	armv6k	armv7e-m	native
armv4t	armv6s-m	armv7ve	
armv5	armv6t2	armv8-a	

Refer ARM-Options[3] for more details about ARM architecture and processors

Please refer Atmel Studio documentation for the supported Atmel ARM devices.

[1] https://launchpad.net/gcc-arm-embedded/5.0/5-2016-q1-update
[2] http://distribute.atmel.no/tools/opensource/Atmel-ARM-GNU-Toolchain/5.3.1/readme.txt
[3] https://gcc.gnu.org/onlinedocs/gcc/ARM-Options.html

6. Contact Information and Disclaimer

6.1 Contact

For support on Atmel ARM GNU Toolchain, visit design support[1].

Users of ARM GNU Toolchain are also welcome to discuss on the AT91SAM Community website[2] forum.

6.2 Disclaimer

Atmel ARM GNU toolchain is distributed free of charge for the purpose of developing applications for Atmel SAM devices. Atmel ARM GNU Toolchain comes without any warranty.

[1] http://www.atmel.com/design-support/
[2] http://www.at91.com/

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**